Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Uma Ferramenta de Geração de Grafos de Proveniência para Bioinformática

Alexandre Lins de Albuquerque Andrade

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientadora
Prof.ª Dr.ª Genaína Nunes Rodrigues

Coorientadora
Prof.ª Dr.ª Maristela Terto de Holanda

Brasília
2013

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Coordenadora: Prof.ª Dr.ª Maristela Terto de Holanda

Banca examinadora composta por:

Prof.ª Dr.ª Genaína Nunes Rodrigues (Orientadora) — CIC/UnB
Prof.ª Dr.ª Aletéia Patrícia Favacho de Araújo — CIC/UnB
Prof.ª Dr.ª Maristela Terto de Holanda — CIC/UnB

CIP — Catalogação Internacional na Publicação

Andrade, Alexandre Lins de Albuquerque.

Uma Ferramenta de Geração de Grafos de Proveniência para Bioinformática / Alexandre Lins de Albuquerque Andrade. Brasília : UnB, 2013.
89 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2013.

1. workflow científico, 2. proveniência de dados, 3. bioinformática, 4. *web*

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Uma Ferramenta de Geração de Grafos de Proveniência para Bioinformática

Alexandre Lins de Albuquerque Andrade

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof.ª Dr.ª Genaína Nunes Rodrigues (Orientadora)
CIC/UnB

Prof.ª Dr.ª Aletéia Patrícia Favacho de Araújo
CIC/UnB

Prof.ª Dr.ª Maristela Terto de Holanda
CIC/UnB

Prof.ª Dr.ª Maristela Terto de Holanda
Coordenadora do Bacharelado em Ciência da Computação

Brasília, 4 de março de 2013

Dedicatória

Dedico esta monografia à minha família, à minha namorada e aos meus amigos, sem os quais a trajetória teria sido muito mais difícil.

Agradecimentos

Em primeiro lugar agradeço aos meus pais, Henrique Andrade e Luciana Andrade, pelo apoio durante toda a graduação.

Agradeço, também, a meus amigos, por estarem comigo nos momentos necessários de descontração;à minha namorada, Talita Soares, que me deu suporte nos momentos difíceis sempre me dando força para continuar.

Agradeço, ainda, à minha orientadora Prof.ª Genaína Nunes Rodrigues e à minha coorientadora Prof.ª Maristela Terto de Holanda pelo apoio e sugestões dados durante o desenvolvimento deste trabalho.

Resumo

A maioria dos experimentos realizados pela bioinformática possui uma grande quantidade de dados e são compostos por várias etapas. Algumas dessas etapas demandam tempo dos pesquisadores – tempo este que poderia ser utilizado de forma mais eficiente. Nessa perspectiva, foram criados *workflows* científicos para possibilitar certa automatização nos processos. Contudo, por se manipular grande quantidade de informação, é também muito importante para os pesquisadores considerar o caminho que esses dados percorrem durante a execução do *workflow*. A persistência do caminho desses dados é chamado de proveniência de dados. Este trabalho de conclusão de curso tratou de criar, a partir de um modelo de dados, um sistema *web* capaz de criar grafos de proveniência a partir da persistência das informações em um banco de dados.

Palavras-chave: workflow científico, proveniência de dados, bioinformática, *web*

Abstract

Vast majority of the experiments realized by bioinformatics uses a great amount of data and follows a series of steps. Some of these steps demands time from the researchers that could be used in a more eficient way. In this perspective, scientific workflows were created to provide certain automation in these steps. However, when using great amount of information,knowing abou the path the data traveled is very important for the researchers. The persistence of the path of this data is called provenance. Therefore, this final project consisted in creating, by using a data model as a starting point, a web system capable of generating provenance graphs from the persistence of the data in a database.

Keywords: scientific workflow, provenance, bioinformatics, web

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A área de bioinformática cresceu devido à necessidade dos biólogos de utilizar e interpretarem grandes volumes de dados gerados a partir de pesquisas na área de genoma. O maior objetivo da área é o desenvolvimento de modelos computacionais (chamados também de modelos *in silico*) que complementem experimentos biológicos realizados [8].

Grande parte dos experimentos realizados *in silico* é de larga escala e composta por vários programas encadeados, com diversas entradas e saídas intermediárias e finais. Assim, a realização desses experimentos sem auxílio de uma ferramenta computacional torna-se muito onerosa, pois, sem ela, cada programa desse fluxo será executado manualmente pelo cientista, fornecendo as saídas de um programa como entradas do programa seguinte. Além de onerosa, oferece riscos, podendo resultar em erros difíceis de localizar e, consequentemente, de corrigir [11].

As ferramentas computacionais que auxiliam os cientistas na realização dos experimentos, propiciando uma maior automação dos trabalhos, são os gerenciadores de *workflows*. São eles que dão suporte à criação dos fluxos de programas encadeados, juntamente com suas entradas e saídas necessárias. Já para acessar o caminho que os dados percorreram, os cientistas utilizam as chamadas ferramentas de proveniência de dados. É ela que capta e armazena o caminho que os dados percorreram, os programas que os executaram, a identificação dos usuários e tudo que ocorreu no processo.Em *workflows*, a proveniência dos dados procura capturar uma descrição completa de todos os passos do fluxo, o que é muito importante para uma posterior verificação. Além disso, a proveniência possibilita a reprodutibilidade, o compartilhamento e a reutilização pela comunidade científica [2][9].

1.1 Problema

Os cientistas da área de bioinformática consideram de extrema importância a utilização de ferramentas de *workflow* com proveniência de dados para o auxílio e suporte em seus trabalhos. Sem elas, muitas vezes os cientistas não tem acesso ao caminho percorrido pelos dados no *workflow* e por isso é necessário refazer todos os passos várias vezes, tomando-lhes um tempo precioso. Nesse contexto, a proposta deste trabalho consiste em implementar um sistema que seja capaz de criar grafos de proveniência a partir dos dados inseridos e resultantes da execução de um *workflow*.

1.2 Hipótese

Utilizou-se como hipótese deste trabalho, que um modelo de proveniência juntamente com um banco de dados para persistência da proveniência iria automatizar e facilitar o processo de pesquisa dos cientistas.

1.3 Motivação

Conseguir uma implementação de um modelo de proveniência a partir de um sistema *web*, automatizando esse processo, pode trazer grande agilização e otimização dos procedimentos de pesquisa no meio científico voltado à bioinformática.

1.4 Objetivos

1.4.1 Objetivo Geral

O objetivo deste presente trabalho é desenvolver um sistema que faça a ligação entre um modelo de proveniência e um modelo de banco de dados, com a possibilidade de exportação dos dados e a criação do grafo de proveniência dos dados gravados.

1.4.2 Objetivos Específicos

Para o desenvolvimento desse trabalho, têm-se os seguintes objetivos específicos:

- Especificação do Modelo Relacional dos dados;
- Especificação e implementação de uma ferramenta *web* para geração de grafos de proveniência;
- Implementação da ferramenta *web* com capacidade para múltiplos usuários e projetos;
- Gerenciamento de entidades, atividades, agentes, coleções e relações em um banco de dados;
- Criação de um arquivo XML, descrevendo o a proveniência dos dados a partir das informações contidas em um banco de dados;
- Criação da imagem de um grafo de proveniência a partir das informações existentes em um banco de dado.

1.5 Estrutura do Documento

Esta monografia está organizada da seguinte forma:

- Capítulo 1: faz uma introdução aos temas, *workflow* e proveniência de dados, bem como especifica o problema, a hipótese, a motivação e os objetivos deste trabalho;

- Capítulo 2: provê a fundamentação teórica necessária para o entendimento do trabalho;
- Capítulo 3: descreve a ferramenta resultante deste trabalho, sua arquitetura, os casos de teste e uma análise comparativa com trabalhos anteriores;
- Capítulo 4: apresenta a conclusão e as sugestões de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

2.1 *Workflow*

Em diferentes áreas os experimentos científicos dependem da aquisição, manipulação e processamento de dados em grande quantidade. Esses experimentos, que na maioria das vezes seguem um mesmo fluxo, são dispendiosos em relação ao tempo e aos custos operacionais. Visando reduzir esses custos e facilitar a repetição do mesmo processo inúmeras vezes, cientistas e pesquisadores passaram a utilizar *workflows*.

Workflow é uma facilitação ou automação computadorizada de um processo, em parte ou como um todo [13]. *Workflow* científico nada mais é que a descrição de uma série de passos que, dado uma entrada, resultam na resolução de um problema no âmbito científico. Muitas vezes esses passos não são simples, possuem muitas dependências uns dos outros e são necessárias análises entre eles [28].

Assim, pode-se considerar o ciclo de vida de um *workflow*(esquematizado na Figura 2.1)conforme segue [17]:

- Hipóteses/Definição dos objetivos

  -Devem ser definidos quais são os objetivos a serem alcançados com a execução do *workflow*, para que ao final, possa haver algum parâmetro para comparação entre os resultados obtidos e aqueles que eram esperados.

- Projeto

  -Na maioria dos casos é orientado a grafos.

  -Além disso, utiliza-se, muitas vezes, *templates* de *workflows* já existentes.

- Preparação do *workflow*

  -A origem dos dados é definida.

  -Parâmetros são incluídos pelo usuário.

- Execução

  -O *workflow* é executado a partir da descrição recebida com os dados de entrada.

- Monitoramento (muito importante em simulações de larga escala)

  -Acontecem eventos que são resultados intermediários do *workflow*.

-Ocorre a execução de serviços que estavam no fluxo.

-Dependendo das informações recebidas, decide-se pela continuidade da execução ou pela interrupção.

- Análise pós-execução

  -Avaliação dos resultados.

  -Verificação do motivo de alguma falha.

  -Comparação dos resultados obtidos com aqueles que eram esperados. Com essa comparação, são definidos novos objetivos e hipóteses, que levam à modificações no projeto do *workflow* para que os resultados sejam os mais próximos possíveis dos esperados.

Figura 2.1: Ciclo de vida de um *workflow* [17].

Nos últimos anos, a ciência empírica vem evoluindo das experimentações físicas para pesquisas baseadas na computação de dados [14]. Um exemplo muito comum é o da bioinformática, que ficou mais conhecida a partir do Projeto Genoma. Nele, repositórios de dados hospedados em grandes instituições do ramo fornecem os dados recolhidos por meio da *Genome-Wide Association Studies* e possibilita que pesquisadores liguem determinados genotipos a uma variedade de doenças [14]. Esses e outros avanços foram obtidos devido à "popularização" dos *workflows* científicos. Com o *workflow*, houve um aumento significativo no rendimento das pesquisas e na confiabilidade dos resultados produzidos, pelo fato que os WfMS(*Workflow Management System*) utilizados são executados em máquinas que possuem grande capacidade de processamento, ou ainda por serem executadas em *grid*.

2.2 *Workflow Management Coalition*

A *Workflow Management Coalition* foi criada por um grupo de organizações com o objetivo de resolver o problema de falta de padronização na construção dos WfMSs. Essas organizações perceberam que todos os sistemas possuíam características comuns, logo, havia uma possibilidade de conexão entre eles. Em consquência, a WFM *Coalition* criou especificações para a implementação de WfMSs, levando em consideração a possibilidade de haver uma interoperabilidade entre diferentes sistemas de *workflow* e que facilitasse a integração com outros serviços de TI(Tecnologia da Informação) [13].

2.3 *Workflow* Científico

De início, importa observar que cientistas executam suas pesquisas com um grau de flexibilidade muito superior ao feito no âmbito de negócios. Um cientista pode facilmente querer filtrar os dados que estão sendo obtidos, por exemplo, de um aparelho medidor e, mesmo que esse filtro não tenha sido planejado originalmente, isso é perfeitamente aceitável em um fluxo de pesquisa [30].

Apesar de a flexibilidade ser uma das maiores propriedades de um *workflow* científico, existem inúmeros processos e experimentos complexos, que precisam seguir regras estritas para que sejam realizados com sucesso. Assim, uma grande diferença entre os *workflows* de negócios e os científicos é o fato de que um *workflow* científico nem sempre está completamente definido antes de seu início [30].

Na área acadêmica, muitos setores utilizam amplamente os *workflows*. Na Astrologia por exemplo, o ambiente Triana foi usado para a visualização de dados a partir de uma simulação da formação de uma galáxia [29]. Na Sismografia, um *workflow* foi usado para construir um ambiente em que análises científicas complexas e de larga escala pudessem ser agendadas. Essas análises foram utilizadas para computar curvas de probabilidades sísmicas e prever a magnitude de abalos sísmicos em uma certa área, e em um determinado período de tempo [10].

Workflows científicos dão suporte e automatizam tarefas que normalmente seriam propensas a repetições e erros. Contudo, sistemas de gerenciamento de *workflows* científicos não são feitos somente para criação e execução dos *workflows*. Áreas como modelagem, análise e reuso de *workflows* também têm-se mostrado muito importantes. Nesse sentido, pode-se afirmar que a utilização de *workflows* tem como objetivos primordiais permitir que cientistas e pesquisadores tenham seu tempo voltado para atividades de seu domínio - não precisando se preocupar com gerenciamento de dados ou problemas com os *softwares* - e otimizar a execução do *workflow* com os recursos disponíveis [17].

2.4 SWfMS - *Scientific Workflow Management System*

SWfMS é um sistema criado, especificamente, para gerenciar *workflows* de âmbito científico. É um sistema que define, modifica, gerencia, monitora e executa *workflows* científicos por meio de uma "tarefa científica", cuja execução é orientada por uma representação computadorizada da lógica computacional do *workflow* [16].

2.4.1 Arquitetura

Foram definidos sete requisitos-chave para a arquitetura de um SWfMS [16], abaixo listadas.

- Interface com o usuário customizável e suporte à interação com o usuário: na maioria dos casos, os cientistas usuários de SWfMS têm pouca ou nenhuma experiência de desenvolvimento de *software*. Tendo isso em vista, a interface utilizada pelo gerenciador de *workflow* deve ser amigável e capaz de ser modificada com facilidade para um direcionamento melhor, dependendo do caso de uso e das necessidades do usuário.
- Suporte à reprodutibilidade: a reprodutibilidade é um princípio fundamental em qualquer método científico. Os resultados obtidos após um *workflow* científico devem poder ser reproduzidos novamente para verificação.
- Integração com diferentes tipos de programas e ferramentas distribuídos: em todo *workflow* cientistas têm a necessidade de integrar vários programas e ferramentas. Muitas vezes, cada um desses programas pode ter sido desenvolvido em uma linguagem de programação diferente e pode ter diversos tipos de chamada. Portanto, a arquitetura do SWfMS deve ter uma abstração para que vários tipos de programas possam ser rodados e isso fique transparente para o usuário.
- Controle distribuído e heterogêneo dos produtos de dados: assim como as ferramentas e programas de um SWfMS, cada *workflow* produz, normalmente, uma grande quantidade de dados de diferentes tipos, formatos e tamanhos. A arquitetura deve prover uma abstração também para esses dados de uma forma que o usuário tenha acesso somente a um "produto de dado".
- Suporte à tecnologia de ponta: hoje em dia, muitos problemas científicos requerem tecnologia de ponta para serem resolvidos, como computação em *grid*, computação em nuvem. Logo, um requisito para a arquitetura é a separação da área de problemas científicos daquela de problemas de tecnologia, para que o sistema esteja pronto quando for necessário incluir novas tecnologias sem afetar a parte científica.
- Monitoramento de *workflow* e tratamento de falhas: principalmente para *workflows* de grande escala, que demoram muito tempo para serem executados por completo, o monitoramento e o tratamento de falhas são muito importantes. Com eles, o usuário pode ter acesso a resultados intermediários antes de que o *workflow* ser executado por completo. E, nos casos em que ocorrem problemas durante essa execução, o *workflow* deve tentar contornar o erro por conta própria antes de ser finalizado, e ainda fazer uma gravação do motivo e local onde esse erro ocorreu para posterior verificação do usuário.
- Interoperabilidade: quanto mais as pesquisas se tornam colaborativas, e os grupos de pesquisa são geograficamente distantes, mais os *workflows* científicos são distribuídos e colaborativos também. Podem ser construídos a partir de vários *subworkflows* e cada um pode ser gerenciado por um SWfMS diferente. A partir disso, a arquitetura deve facilitar essa interoperabilidade entre SWfMSs para que todos os *workflows* possam tirar proveito das ferramentas e programas incluídas em todos os outros SWfMS.

2.5 Proveniência de Dados

Proveniência de dados, também chamada de linhagem do dado, é a descrição da origem de um dado e do processo pelo qual ele chegou ao destino final [5]. Dizer que o "processo" também faz parte da proveniência de dados significa que tudo o que acontecer com esse dado, desde sua criação ou inserção deve ser descrito. Isso inclui o caminho que ele percorreu e os processos intermediários que agiram sobre ele até ser considerado produto final.

A proveniência de dados já foi muito estudada tanto no âmbito de bancos de dados quanto no de *workflows*. Em banco de dados foi definida na maioria das vezes, para objetos relacionais ou muito complexos.

No contexto de *workflows* científicos, existem duas formas distintas de proveniência de dados [7]:

- Proveniência prospectiva: captura a especificação de uma tarefa computacional, como por exemplo *workflow*. Corresponde aos passos que precisam ser seguidos para gerar um produto de dado ou uma classe de produtos de dado;
- Proveniência retrospectiva: captura os passos que foram executados, assim como, as informações sobre o ambiente de execução usado para derivar um produto de dado específico. É um registro detalhado da execução de uma tarefa computacional.

Uma parte muito importante na proveniência de dados são informações sobre os parâmetros e os dados de entrada fornecidos ao *workflow*, pois eles, juntamente com as proveniências prospectiva e retrospectiva, são úteis para reproduzir e validar os processos.[7]

Atualmente existem muitos modelos distintos de proveniência de dados, porém todos têm como objetivo a gravação dos dados da proveniência para que de alguma forma, posteriormente, possam ser recuperados e utilizados. Alguns deles, de maior relevância no âmbito da bioinformática e mais utilizados na literatura são apresentados nas seções seguintes.

2.5.1 Modelo W7

Esse modelo é baseado na ontologia de Bunge, que tem como objetivo descrever as propriedades de um objeto de caráter geral. O modelo leva esse nome porque a proveniência é estruturada por meio das respostas a sete perguntas: O quê?, Quem?, Quando?, Onde?, Como?, Qual?, e Por quê?(Em inglês: *What?, Who?, When?, Where?, How?, Which? e Why?*). A visão geral do modelo W7 pode ser vista na figura a seguir, que mostra em qual fase cada uma das sete perguntas é respondida.

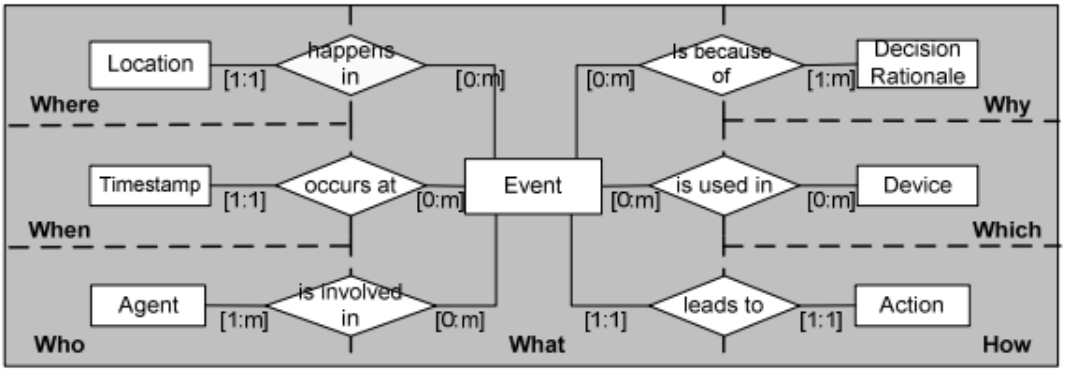

Figura 2.2: Visão geral do modelo W7 [26].

O ponto positivo desse modelo é o fato de que as perguntas geram a maioria das respostas necessárias para se ter uma boa proveniência do dado. Seu lado negativo é a complexidade da representação gráfica apresentada pelo modelo.

2.5.2 *Provenance Vocabulary*

É voltado para a proveniência de dados publicados na *web*. Destacam-se as três classes que dão a estrutura principal do modelo (conforme indica a Figura 2.3):

- Artefatos: podem ser tanto entradas como produtos. Possuem três subclasses, *DataItem*(dado do artefato), *File*(conjunto de *DataItem*) e *CreationGuideline*)(regras para criação do artefato).

- Execução: é a representação da execução completa de um processo. É composta de duas subclasses, *DataCreation*(processo de criação do dado) e *DataAccess*(processo pelo qual o dado foi acessado).

- Atores: representa uma entidade ativa que pode afetar a execução das ações ou dos processos. Possui duas subclasses, *HumanActor* (pessoa ou organização) e *NonHumamActor* (entidade não humana).

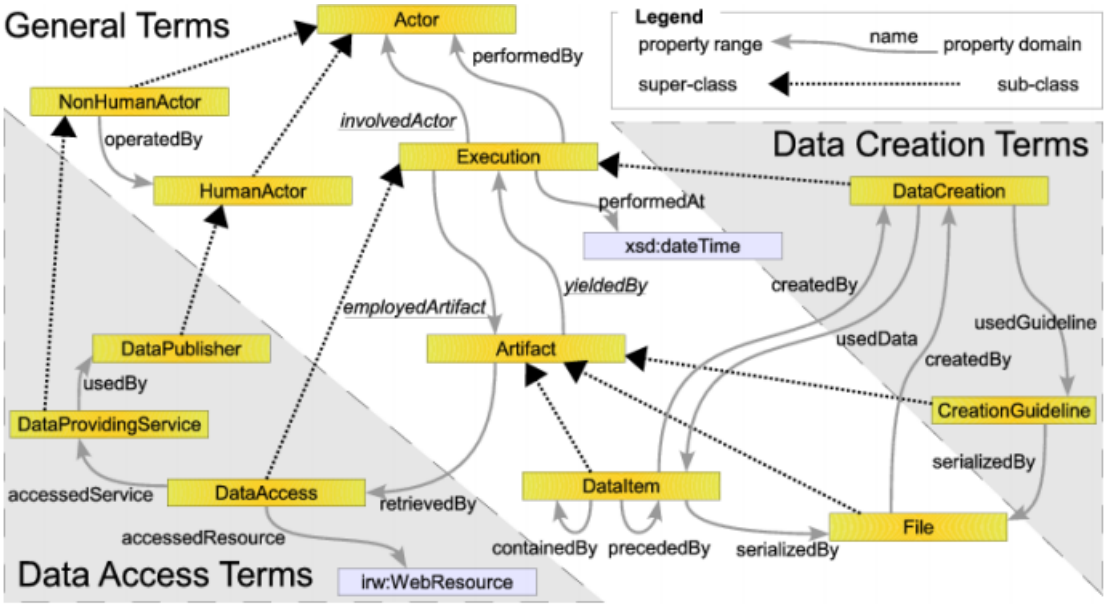

Figura 2.3: Classes e propriedades do modelo *Provenance Vocabulary* [12].

2.5.3 *Provenir Ontology*

Provenir Ontology é um modelo de proveniência de dados genérico com alta capacidade de adaptação em diferentes aplicações. Para melhor entendimento do modelo, Sahoo descreve dois conceitos que podem representar as classes de metadados (também esquematizadas na Figura 2.4):

- Ocorrentes: são entidades que ocorrem em sucessivas fases temporais. Suas classes são Dado (classe que representa tanto o material inicial, intermediário e o produto final de um experimento científico) e Agente (classe que afeta o processo individual).

- Contínuos: são entidades que permanecem ou continuam a existir através do tempo, mesmo quando passam por vários tipos de mudanças. Sua classe é Processo (classe que demonstra uma ação que afeta os dados).

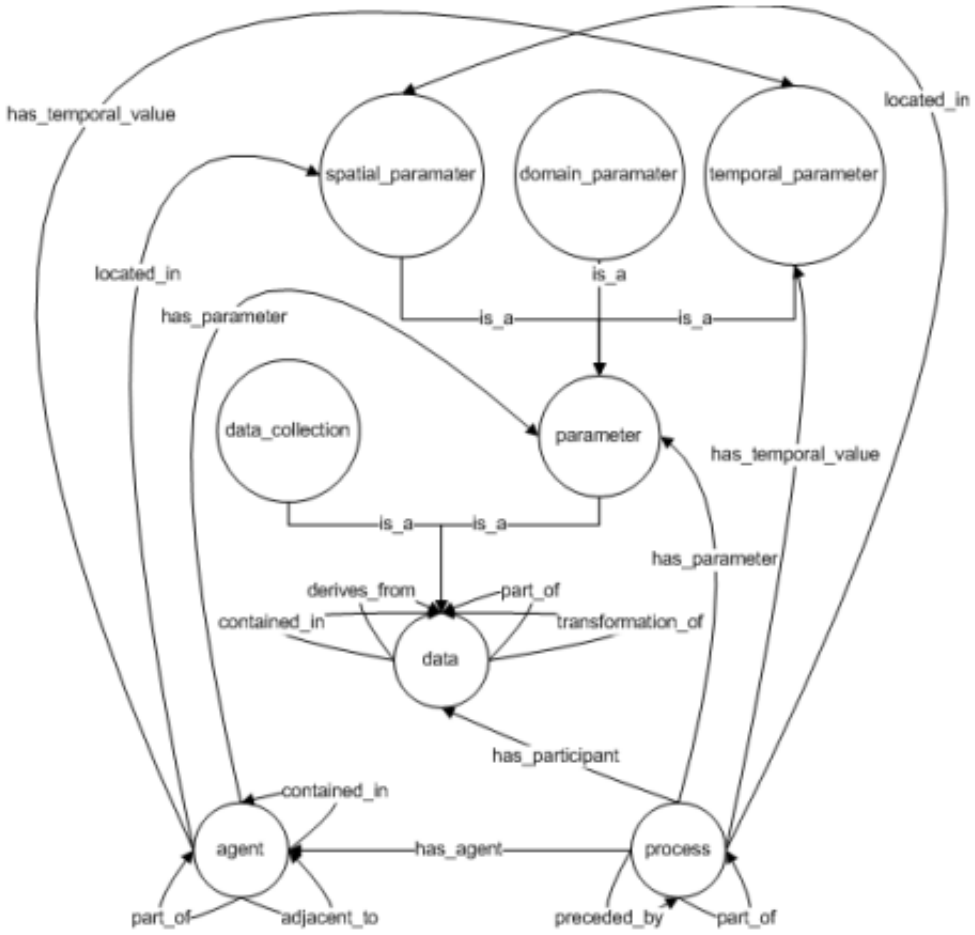

Figura 2.4: Classes e seus relacionamentos no modelo *Provenir Ontology* [27].

2.5.4 *Open Provenance Model*(OPM)

O OPM é o resultado do *Provenance Challenge*, iniciado em 2006 no primeiro IPAW (*International Provenance and Annotation Workshop*). Desde o início, os autores originais tinham a intenção de criar um modelo de dados aberto do ponto de vista da interoperabilidade, que também recebesse contribuições da sua comunidade, usuários e revisores. Para garantir essa condição, foi criada a versão 1.1 do OPM que visava ser um modelo de governança em uma versão de "código aberto" [20]. Os principais objetivos do OPM são [25]:

- Permitir a troca de informações entre sistemas por meio de uma camada de proveniência;
- Permitir um compartilhamento de ferramentas e conhecimento entre os desenvolvedores;
- Fornecer uma representação digital de proveniência para "qualquer coisa".

Assim como são definidos os objetivos, define-se também o que não faz parte dos objetivos do OPM:

- Definir a forma de armazenamento e manipulação das informações internas;
- Definir a linguagem na qual será desenvolvido o modelo;
- Definir protocolos, tanto de armazenamento das informações de proveniência em repositórios, como para consulta em repositórios de proveniência.

O modelo básico do OPM possui três tipos de nós, e cinco tipos de elos (dependências).

Nós

Na Figura 2.5, pode-se observar as representações gráficas dos nós desse modelo. A imagem à esquerda corresponde ao artefato, um objeto em um espaço de tempo específico, que pode ser usado ou gerado por um processo. A imagem ao centro corresponde ao processo, que simboliza uma ou mais ações executadas utilizando ou gerando um artefato. A imagem à direita corresponde ao agente, uma entidade, dentro de um escopo, que age como catalisador em um processo, afetando a execução do mesmo.

Figura 2.5: Representação gráfica dos nós do modelo OPM [20].

Dependências

- *Used*: representado na (Figura 2.6), indica se um determinado artefato foi usado por um processo;

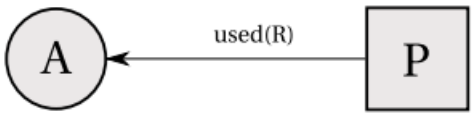

Figura 2.6: Representação gráfica da dependência de uso do modelo OPM [20].

- *WasGeneratedBy*: representado na (Figura 2.7), indica se um artefato foi gerado por um processo específico;

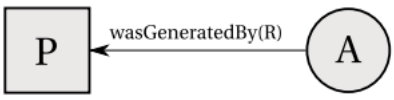

Figura 2.7: Representação gráfica da dependência de geração do modelo OPM [20].

- *WasControlledBy*: representado na (Figura 2.8), indica se um processo foi controlado por um agente;

Figura 2.8: Representação gráfica da dependência de controle do modelo OPM [20].

- *WasTriggeredBy*: representado na (Figura 2.9), indica se um Processo P1 foi acionado por outro Processo P2;

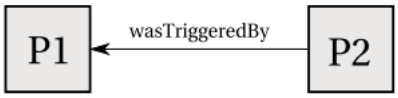

Figura 2.9: Representação gráfica da dependência de acionamento do modelo OPM [20].

- *WasDerivedFrom*: representado na (Figura 2.10), indica se um artefato A1 é derivado de um artefato A2;

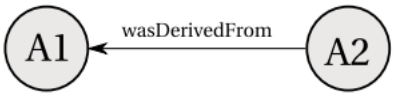

Figura 2.10: Representação gráfica da dependência de derivação do modelo OPM [20].

2.5.5 PROV-DM

O PROV-DM é o modelo de dados conceitual que forma a base da proveniência do W3C (*World Wide Web Consortium*) [3]. Sua primeira versão foi desenvolvida em outubro de 2011 e sua última versão publicada em maio de 2012. Mesmo sendo um modelo muito recente, já é largamente utilizado uma vez que usa os mesmos princípios do OPM. O PROV-DM é mais detalhado, o que permite que a proveniência seja demonstrada de uma forma mais precisa. Este modelo está organizado em seis componentes [3] conforme segue:

- Entidades e atividades: entidades podem representar qualquer objeto e atividades representam os processos que usam e geram entidades;
- Derivações: descrevem relações entre diferentes entidades durante a transformação executada pelas atividades, o que permite que sejam demonstradas as dependências geradas e usadas das entidades;
- Agentes e Responsabilidades: agentes são entidades que têm algum papel na execução das atividades. Eles recebem também atribuições de outros agentes e podem ter direitos sobre outras entidades;
- Alternativo: emite uma conexão entre entidades que se referem à mesma coisa;

- Proveniência de proveniência: cria-se um conjunto de descrições de proveniência, que é uma entidade, permitindo que a proveniência de proveniência seja expressada;
- Coleções: é uma entidade que fornece uma estrutura para seus membros, que também são entidades. Esses membros são chamados "membros da coleção".

Tipos
São definidos dois tipos e quatro subtipos que dão origem aos nós do grafo:

- Atividade: representa os possíveis processos executados que deram origem ao objeto foco da proveniência;
- Entidade: representa qualquer objeto que seja capaz de representar uma proveniência.

  Os subtipos gerados pelos tipos descritos acima são:

  - Agente: representa qualquer entidade que exerce ação sobre uma atividade ou possua responsabilidade sobre uma Entidade;
  - Coleções: conjunto de entidades;
  - Conta: conjunto de informações que compõem um grafo de proveniência;
  - Plano: conjunto de ações que um agente deve seguir.

Os símbolos utilizados para representar os nós do grafo são agente, atividade e entidade (que representa também os tipos coleção e plano), como mostra a Figura 2.11:

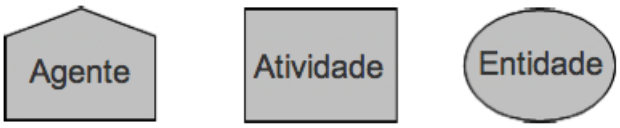

Figura 2.11: Representação gráfica dos tipos do modelo PROV-DM [3].

Tendo em vista que um dos maiores objetivos do PROV-DM é se tornar uma recomendação W3C, foram produzidas diversas especificações que dão apoio ao modelo. Elas são [25]:

- PROV-DM [22]: visão geral das principais características do modelo PROV-DM;
- *PROV-CONSTRAINTS* [6]: define um conjunto de restrições do modelo.;
- PROV-N [23]: define uma notação para a proveniência destinada ao uso em linguagem descritiva;
- PROV-O [15]: define uma ontologia OWL-RL que permite mapear o modelo PROV-DM para o padrão RDF;
- PROV-AQ [21]: define mecanismos de acesso e consulta de proveniência;
- *PROV-PRIMER* [4]: mostra uma introdução ao modelo;

- *PROV-SEM* [24]: define uma semântica formal do modelo;
- PROV-XML [19]: esquema XML para o modelo.

Assim como no OPM, no PROV-DM existem relações entre os "objetos" do modelo. Elas são representadas pelas arestas do grafo de proveniência e demonstram as relações entre os nós adjacentes.

Como o modelo a ser usado neste trabalho será o PROV-DM, e por isso foi feita uma análise mais aprofundada deste.

Capítulo 3

ProvBioUnB

O sistema ProvBioUnB é o projeto *web*, resultado desta monografia, criado com base no trabalho de Paula [25]. Nele é feito o gerenciamento das entidades, das atividades, das agentes, das coleções e das relações que servirão como base para a criação de um grafo de proveniência.

3.1 Por que usar Interface *web*

Aplicação *web* é uma aplicação padrão, porém, ela é acessada por meio da Internet, utilizando algum navegador como cliente. A grande vantagem de se usar uma interface *web* é a facilidade de se para fazer atualização nos sistemas sem a necessidade de que essa atualização seja instalada em cada computador que a utiliza.

Outra vantagem é a mobilidade que um sistema *web* pode oferecer. Mesmo em sistemas que necessitam de grande segurança, construído de forma correta, é possível que a aplicação seja acessada de qualquer lugar onde haja acesso a tal rede, diferentemente de aplicações locais, que só são acessíveis localmente. O custo de infra-estrutura de aplicações *web* é normalmente inferior a outros tipos de aplicações semelhantes.

3.2 Requisitos

Os requisitos do sistema podem ser divididos em duas partes, sendo a primeira delas referente ao servidor que hospeda o sistema, e a segunda diz respeito ao cliente (usuário do sistema).

Para que o Proveniência UnB tenha todas suas funcionalidades executadas corretamente é necessário que o servidor onde o programa será instalado (que pode ser um servidor local ou não) possua PHP 5.0 ou superior instalado, e também um banco de dados MySQL. No que se refere ao usuário, basta que este possua acesso à internet e, claro, credenciais de acesso ao programa, que devem ser criadas pelo administrador do sistema.

3.3 Aquitetura

O programa desenvolvido para esta monografia utilizou a arquitetura MVC (*Model-View-Controller*). O paradigma MVC é uma forma de dividir a interface da aplicação, ou de um pedaço de uma aplicação em três partes: Modelo, Visão, e Controladora. A Tabela 3.1 mostra o mapeamento das ações executadas pelo sistema com relação à arquitetura MVC.

Tabela 3.1: Correspondência das ações executadas com o modelo MVC [18].

Ação	**MVC**
Entrada	Controladora
Processamento	Modelo
Saída	Visão

Modelo

- É a representação de um dado ou ainda de uma atividade em forma de objeto. Exemplo: uma tabela do banco de dados;
- Faz a gerência dos dados da aplicação, e é responsável por todo tratamento dos dados e de mudança de estado dos objetos;
- Representa as regras de negócio da aplicação.

Visão

- É a forma de visualização do estado representado pela Modelo;
- É responsável pela visualização gráfica/textual dos componentes;
- Renderiza o conteúdo presente na Modelo e define como esse dado será mostrado ao usuário.

Controladora

- Provê facilidades para modificar o estado da Modelo;
- É através da Controladora que o a Visão se conecta com a Modelo, aceitando as entradas do usuário e modificando a Modelo com esses dados.

A Figura 3.1 permite visualizar como é a interação entre os três componentes da arquitetura e o usuário.

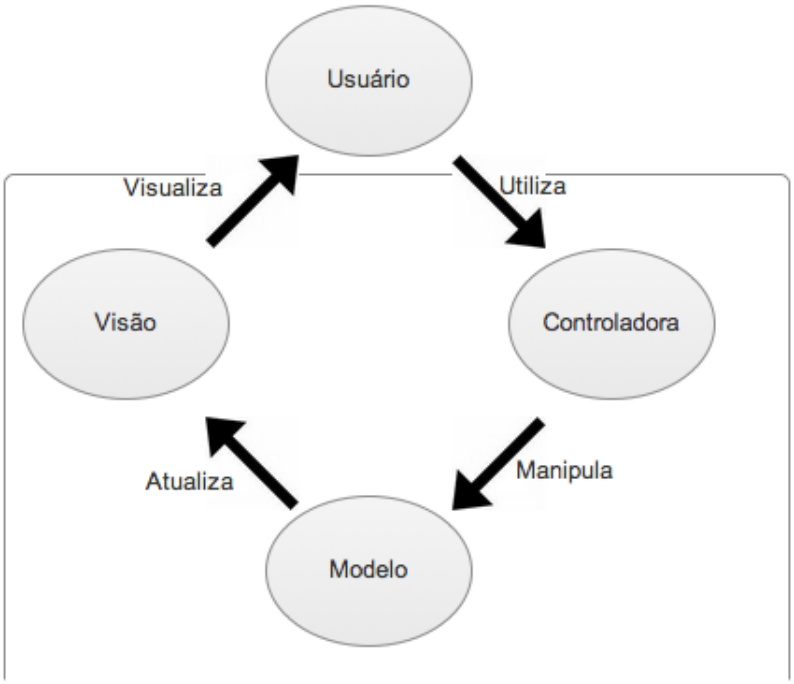

Figura 3.1: Interação MVC-usuário [18].

A arquitetura de informação do sistema *web* foi criada levando em consideração os tipos de usuários que o utilizarão. Identificou-se a estrutura adequada para o sistema, a melhor navegação e a organização adequada para o conteúdo proposto. Para a criação da arquitetura foi feito um protótipo semi-funcional a partir do qual foi possível identificar as necessidades do sistema. Esse protótipo serviu de base para a criação do sistema real. Algumas modificações foram feitas para a versão final após verificar que o funcionamento não ocorreu como o esperado.

3.3.1 Protótipo

Em sistemas *web* geralmente são construídos protótipos não funcionais, também conhecidos como *blueprints*, antes do desenvolvimento do programa funcional. Isso ocore para que os desenvolvedores e arquitetos de informação possam ter ideias criativas quanto ao *layout* do sistema e quanto à forma como se dará o fluxo de informação do mesmo. Protótipos *web* normalmente não possuem fontes estilizadas, cores ou mecanismos gráficos, já que seu foco maior é a funcionalidade do sistema. Assim, o protótipo foi construído em HTML e prevê todas as funcionalidades presentes no sistema e ainda algumas que estão descritas no capítulo seguinte, para trabalhos futuros. Observando-se a Figura 3.2, que mostra uma tela de listagem de agentes, pode-se perceber que a base do protótipo foi mantida para o sistema final, entretanto, o *menu* superior foi redirecionado para a esquerda por motivos de hierarquia da informação.

Figura 3.2: Tela de listagem de agentes do protótipo.

As opções de menu que estão localizadas na parte superior do protótipo foram postas no *menu* da esquerda do produto final. Isso ocorreu porque dois *menus* separados dariam a impressão de algum tipo de hierarquia entre eles, que neste caso, não existe.

3.4 Modelo de Dados

Para a criação do modelo de dados deste trabalho, usou-se como base a Tabela 3.2. Nela, são descritas as mínimas informações necessárias para proveniência de projetos de bioinformática. A partir dessa tabela, foi criado o modelo relacional que posteriormente deu origem ao banco de dados do ProvBioUnB. Cada elemento da tabela foi transformado em uma tabela do banco de dados e as colunas de cada uma dessas tabelas foram baseadas nas informações mínimas necessárias para a correta utilização do PROV-DM.

Tabela 3.2: Informações do grafo de proveniência para projetos de bioinformática [25].

PROJETO		**CONTA**	
Nome	Nome do projeto	Nome	Nome do experimento
Descrição	Descrição do projeto	Descrição	Descrição do experimento
Instituições Financiadoras	Lista das instituições que financiaram o projeto	Local	Local de execução
		Data Inicial	Data inicial da execução
Instituições Participantes	Lista das instituições que participam do projeto	Data Final	Data final da execução
		Versão e Data	Número e data da versão gravada
Coordenador	Nome do coordenador do projeto	Anotações	Qualquer informação adicional sobre o experimento
Data Inicial	Data de início do projeto		
Data Final	Data final do projeto		
AGENTE		**ATIVIDADE**	
Nome	Nome do usuário	Nome	Nome da atividade
Instituição	Instituição do usuário	Programa	Nome do programa
Cargo	Cargo ou função	Versão	Versão do programa
Função	Função no experimento	Comando	Linha de comando com os parâmetros utilizados
Grupos	Grupos para filtrar o grafo de proveniência	Função	Descrição do que a atividade executou
Anotações	Qualquer informação adicional sobre o agente	Hora Inicial	Data e hora em que a atividade foi iniciada
		Hora Final	Data e hora em que a atividade foi concluída
		Ambiente	Descrição do ambiente computacional em que a atividade rodou
		Grupos	Grupos para filtrar o grafo de proveniência
		Anotações	Qualquer informação adicional sobre a atividade
COLEÇÃO		**ENTIDADE**	
Nome	Nome da coleção	Nome	Nome da entidade
Tamanho	Número de entidades contidas na coleção	Descrição	Descrição do conteúdo da entidade
Descrição	Descrição do conteúdo da coleção	Localização	Localização do arquivo ou banco de dados que guarda a entidade
Localização	Localização do arquivo ou banco de dados que guarda a coleção	Grupos	Grupos para filtrar o grafo de proveniência
Grupos	Grupos para filtrar o grafo de proveniência	Anotações	Qualquer informação adicional sobre a entidade
Anotações	Qualquer informação adicional sobre a coleção		

A aplicação desenvolvida neste trabalho foi feita em banco de dados MySQL. O modelo relacional que diz respeito ao projeto pode ser visto na Figura 3.3.

Cada caixa do modelo relacional corresponde a uma tabela e suas colunas são descritas dentro desta caixa, especificando o tipo de cada uma dessas colunas bem como seus tamanhos. Quase todas as tabelas estão relacionadas com a tabela *tb projects* para que possam existir múltiplos projetos criados no sistema, sem que os objetos de um deles apareçam nos outros.

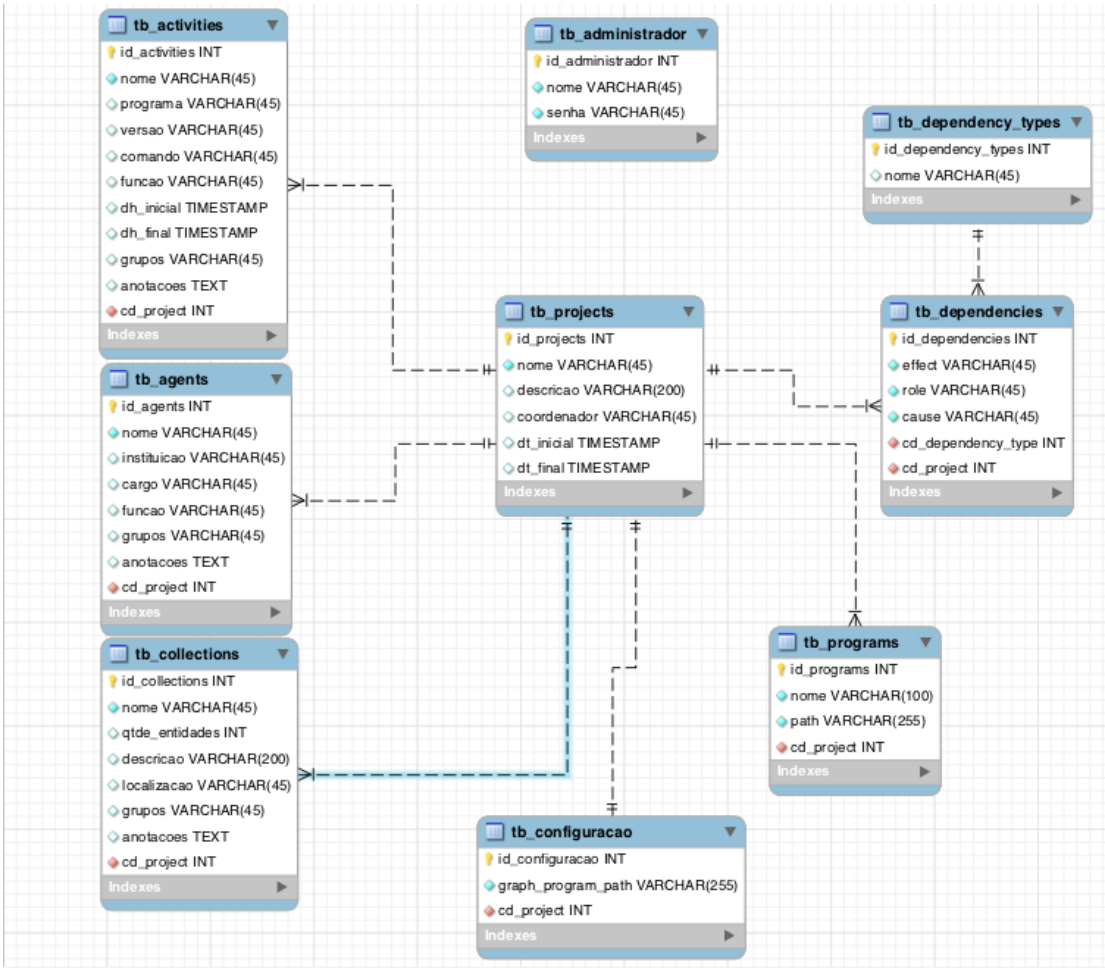

Figura 3.3: Modelo relacional da aplicação.

As tabelas do modelo relacional são descritas a seguir:

3.4.1 Tabelas

- *tb activities*: reúne as atividades que ocorreram no *workflow*.
- *tb agents*: reúne todos os agentes que tiveram participação no *workflow*.
- *tb collections*: reúne as coleções do *workflow*.
- *tb configuracao*: reúne o caminho para o programa gerador de grafo. Cada projeto tem uma entrada nesta tabela.

- *tb programs*: reúne os programas a serem incluídos no *workflow*. Esta tabela só será utilizada em trabalhos futuros, pois a funcionalidade de projeto do *workflow* não foi implementada.
- *tb dependecies*: reúne as dependências entre os objetos do *workflow*. Cada entrada nesta tabela será uma aresta do grafo de proveniência construído.
- *tt dependency types*: é uma tabela auxiliar, em que são incluídos os tipos de dependências permitidos no *workflow*. Para este projeto, essa tabela foi populada com as seguintes entradas: *Used*, *Generated*, *Associated* e *Derived*.
- *tb administrador*: reúne os usuários cadastrados no sistema. O usuário só tem acesso após estar cadastrado nesta tabela.
- *tb projects*: Reúne os diferentes projetos criados.

3.4.2 Relação entre o Modelo de Dados e a Arquitetura

Como explicado anteriormente, a parte Modelo da arquitetura pode representar as tabelas do banco de dados, e de fato, neste projeto, ocorre dessa maneira, como pode ser observado na Figura 3.4.

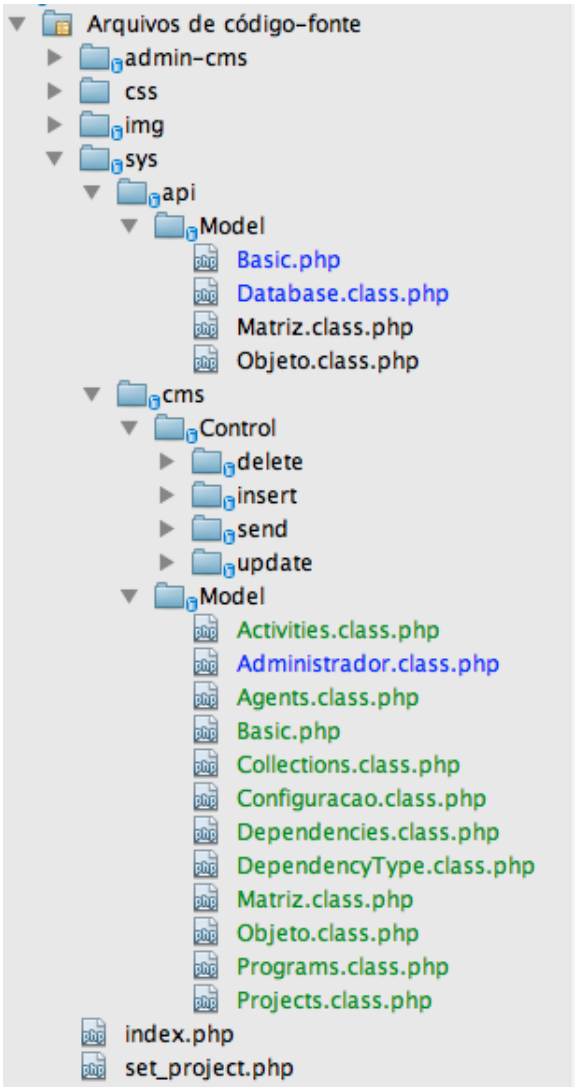

Figura 3.4: Divisão dos arquivos do projeto.

Os arquivos de código-fonte do sistema foram todos colocados dentro da pasta *sys*. Dentro desta pasta, os arquivos foram divididos entre *api* (classes criadas para facilitar o tratamento dos objetos e as consultas ao banco de dados) e *cms* (*Content Management System*, a parte para gerência de todo o conteúdo do sistema).

Na pasta *cms*, existe a divisão entre a parte Modelo e a parte Controladora. A parte Modelo possui as classes que descrevem as tabelas do banco de dados. Como exemplo, a Figura 3.5 mostra a classe Agents.class.php. Nela podemos ver, que assim como em todas as outras classes, suas propriedades correspondem às colunas do banco de dados, mantendo assim o padrão MVC.

```php
<?php

    class Agents extends Objeto{
        public $id_agents;
        public $nome;
        public $instituicao;
        public $cargo;
        public $funcao;
        public $grupos;
        public $anotacoes;
        public $cd_project;

        public $nome_tabela = "tb_agents";
    }

?>
```

Figura 3.5: Código-fonte da classe *Agents*, que descreve a tabela *tb agents*.

Na pasta *Control*, estão os arquivos que irão fazer a conexão entre a Visão e a Modelo. Assim, por exemplo, se um usuário remover algum agente, esta ação irá requisitar um arquivo dentro da pasta *cms/Control/delete*, que irá utilizar o objeto da classe Agents.class.php para remover o agente do banco de dados.

3.5 Descrição do ProvBioUnB

A montagem, o tratamento de dados e a codificação da ferramenta foram realizados levando em consideração os padrões W3C (*World Wide Web Consortium*), que primam pela universalização de acesso por diferentes navegadores e plataformas.

O sistema comporta vários projetos simultâneos, e cada um deles pode conter o seu conjunto de objetos para criação do grafo de proveniência. Na Figura 3.6, pode-se observar a tela em que o usuário seleciona o projeto a ser aberto.

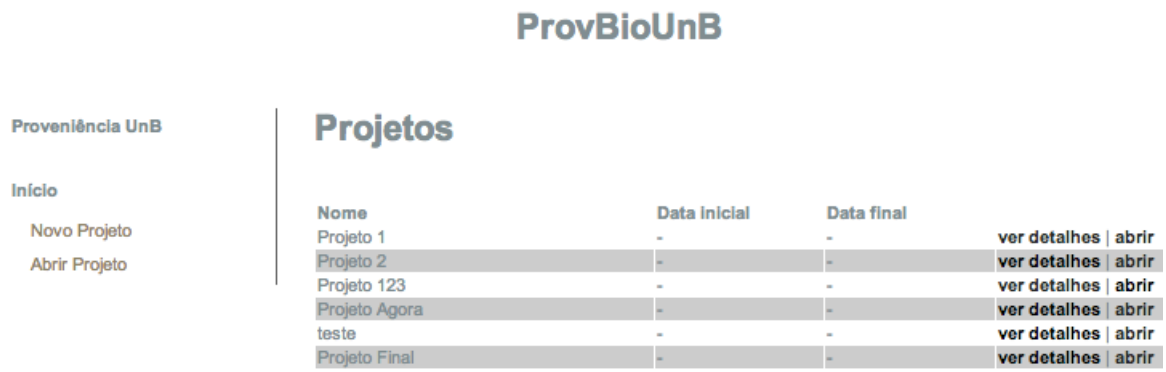

Figura 3.6: Lista de projetos disponíveis para o usuário.

A partir da escolha do projeto pelo usuário, este é direcionado à tela de listagem de agentes, que possui também *links* para as listagens dos outros objetos do sistema.

Figura 3.7: Tela de listagem de agentes, com o menu lateral aparente.

Foram construídos formulários de criação e edição de cada um dos objetos necessários para a criação do grafo de proveniência, no qual encontram-se verificações de existência para os campos obrigatórios no banco de dados. A Figura 3.8 exemplifica um desses formulários. Nela, o asterisco indica que o campo é obrigatório, e o sistema não prossegue com a inserção, caso os campos marcados não sejam preenchidos.

Figura 3.8: Formulário de inserção de um novo agente.

Cada um dos objetos (Agentes, Coleções, Atividades e Relações) pode ser acessado por meio do *menu* lateral dentro da área "Configurações". O formulário responsável pela

inserção das relações entre os objetos do sistema pode ser visto na Figura 3.9. Nesse formulário, define-se a causa e o efeito, bem como qual foi o papel daquela relação e qual o tipo dela.

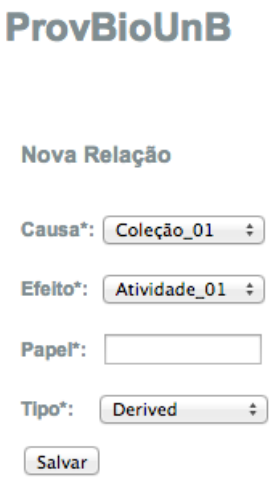

Figura 3.9: Formulário de inserção de relações.

O menu lateral como um todo pode ser visto na Figura 3.10. Algumas das outras opções deste *menu* são descritas mais adiante.

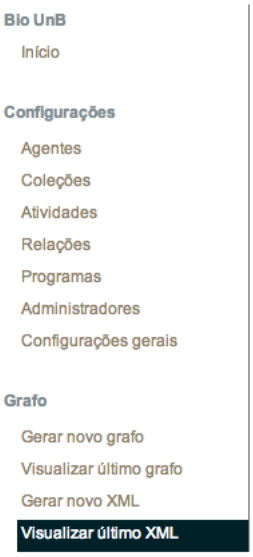

Figura 3.10: Menu lateral do sistema.

Ainda no menu lateral, dentro da área "Grafo", o usuário pode solicitar a geração de um novo grafo, a partir dos dados gravados no banco de dados. Toda vez que essa opção é selecionada, o sistema colhe os dados que estão persistidos naquele momento, e com eles,

cria um arquivo com extensão *dot*[1]. Após a criação do arquivo de descrição do grafo, o sistema transforma essa descrição em uma imagem, onde os nós seguem a representação gráfica do PROV-DM mostrada na Figura 2.11. Um exemplo de grafo gerado pelo sistema pode ser visto na Figura 3.11. Nele, as arestas tracejadas representam relações dos tipos *associated*, *generated* e *used*, e as arestas normais representam relações do tipo *derived*.

Caso o usuário queira apenas visualizar o último grafo gerado, basta selecionar a opção "Visualizar último grafo". Com essa opção, o sistema irá simplesmente abrir o arquivo anteriormente gerado, que já está armazenado na pasta do projeto, sem necessidade de uma nova criação.

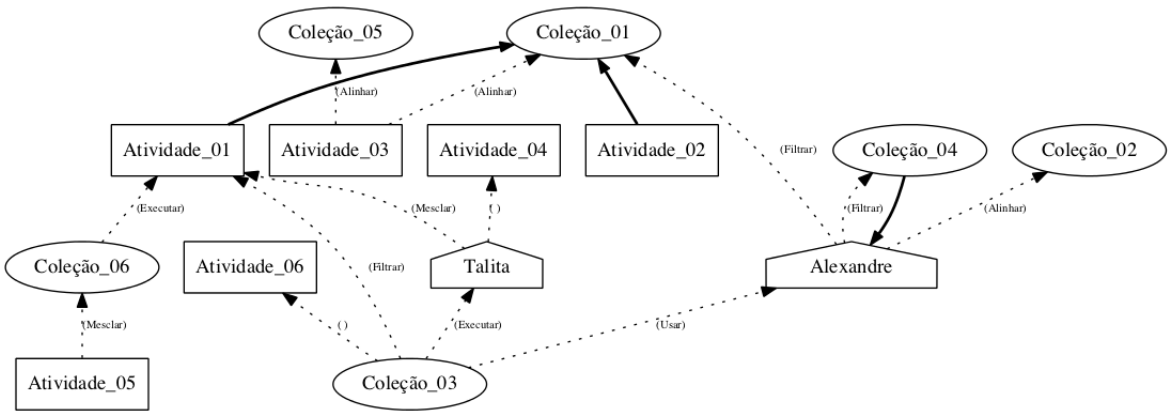

Figura 3.11: Exemplo de grafo gerado pelo sistema.

Caso o usuário queira criar um documento XML com as informações presentes no banco de dados, seleciona a opção "Gerar novo XML". A partir desta ação o sistema realiza as consultas necessárias ao banco de dados e cria um arquivo XML no padrão W3C [19]. Ao utilizar esse padrão, o arquivo poderá ser importado em diferentes programas a critério do usuário. Um exemplo de arquivo XML gerado pode ser visto na Figura 3.12.

[1] *DOT* é uma linguagem de descrição de grafos com gramática definida [1]

```
▼<ns2:opmGraph xmlns:ns2="http://openprovenance.org/model/opmx#" xmlns:xsi="h
 xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:ns5="http://example.com/"
  ▶<ns2:accounts>...</ns2:accounts>
  ▼<ns2:processes>
    ▼<ns2:process id="A001_Família13_Filtro">
       <ns2:account ref=""/>
      ▶<ns2:annotations>...</ns2:annotations>
       <ns2:label value="A001_Família13_Filtro"/>
       <ns2:annotation xsi:type="ns2:Type" value="Script"/>
       <level>0</level>
     </ns2:process>
    ▶<ns2:process id="A002_Família57_Filtro">...</ns2:process>
    ▶<ns2:process id="A003_Família_13_Mesclagem">...</ns2:process>
    ▶<ns2:process id="A004_Família57_Mesclagem">...</ns2:process>
    ▶<ns2:process id="A005_Família13_Alinhamento">...</ns2:process>
    ▶<ns2:process id="A006_Família57_Alinhamento">...</ns2:process>
    ▶<ns2:process id="A007_Família13_Gráfico">...</ns2:process>
    ▶<ns2:process id="A008_Família57_Gráfico">...</ns2:process>
   </ns2:processes>
  ▼<ns2:artifacts>
    ▶<ns2:artifact id="C001_Família13">...</ns2:artifact>
    ▶<ns2:artifact id="C002_Família57">...</ns2:artifact>
    ▶<ns2:artifact id="C003_Família13_Filtrado">...</ns2:artifact>
    ▶<ns2:artifact id="C004_Família57_Filtrado">...</ns2:artifact>
    ▶<ns2:artifact id="C005_ORFS">...</ns2:artifact>
    ▶<ns2:artifact id="C006_Família13_ORFS">...</ns2:artifact>
    ▶<ns2:artifact id="C007_Família57_ORFS">...</ns2:artifact>
    ▶<ns2:artifact id="C008_Família13_Alinhamento">...</ns2:artifact>
    ▶<ns2:artifact id="C009_Família57_Alinhamento">...</ns2:artifact>
    ▶<ns2:artifact id="C010_Família_13_Gráfico">...</ns2:artifact>
    ▶<ns2:artifact id="C011_Família_57_Gráfico">...</ns2:artifact>
   </ns2:artifacts>
  ▼<ns2:agents>
    ▶<ns2:agent id="AG001_João">...</ns2:agent>
    ▶<ns2:agent id="AG002_Tainá">...</ns2:agent>
   </ns2:agents>
  ▶<ns2:dependencies>...</ns2:dependencies>
   <ns2:annotations></ns2:annotations>
 </ns2:opmGraph>
```

Figura 3.12: Exemplo de arquivo XML gerado pelo sistema.

3.6 Casos de Teste

Para averiguar a consistência e garantir que os requisitos foram corretamente atendidos, utilizou-se alguns casos de teste. Todos os casos de teste foram feitos em um MacBook Pro 2.4 GHz Intel Core i5, com 4 GB 1333MHz DDR3 de memória RAM, rodando OS X versão 10.8.2.

Formulários

Para verificar o correto funcionamento de todos os formulários do sistema, foi realizado o caso de teste descrito na Tabela 3.3. Em cada um deles, foi feita a tentativa de submetê-lo sem preencher nenhum campo e, após a mensagem de erro, foi feita a submissão com os campos obrigatórios preenchidos.

Tabela 3.3: Caso de teste para formulários.

Pré-condições	Estar logado
Instrução	Enviar formulário sem preencher nenhum campo
Resultado esperado	Mensagem de erro "Por favor preencha o <campo obrigatório>."
Instrução	Enviar o formulário preenchendo os campos obrigatórios
Resultado esperado	Redirecionar para a listagem dos objetos com o novo objeto em último lugar

Geração de grafo

A Tabela 3.4 corresponde ao caso de teste realizado para a verificação da correta geração do grafo de proveniência. No momento em que a opção "Gerar novo grafo"for selecionada, o sistema deve criar um novo arquivo com a extensão *png*, caso ainda não exista, com uma imagem do grafo gerado a partir das informações presentes no banco de dados.

Tabela 3.4: Caso de teste para geração de grafo.

Pré-condições	Estar logado e ter conteúdo cadastrado
Instrução	Clicar na opção "Gerar novo grafo"
Resultado esperado	Criação(caso não exista) de um arquivo com extensão .png do grafo de proveniência

Geração de arquivo XML

O caso de teste descrito na Tabela 3.5 trata da verificação da correta geração do arquivo XML correspondente ao grafo de proveniência. Quando a opção "Gerar novo XML"é selecionada, o sistema deve criar um novo arquivo XML, caso ainda não exista, com as informações do banco de dados, corretamente hierarquizados de acordo com o padrão de arquivos XML do PROV-DM.

Tabela 3.5: Caso de teste para geração de arquivo XML.

Pré-condições	Estar logado e ter conteúdo cadastrado
Instrução	Clicar na opção "Gerar novo XML"
Resultado esperado	Criação(caso não exista) de um arquivo com extensão .xml com as informações cadastradas no banco de dados

3.7 Análise Comparativa

3.7.1 Persistência de dados

O modelo de dados foi baseado no modelo criado por Paula et al. A grande diferença aparece na forma de persistência utilizada. No trabalho usado como base, toda a persistência foi feita em arquivos XML. Os dados inseridos em um arquivo desse tipo são organizados de forma hierárquica e é dessa forma que entende-se a relação entre os dados. Já no presente trabalho, a persistência dos dados foi feita em um banco de dados relacional. A utilização de bancos de dados em vez de arquivos XML para persistência oferece várias vantagens. Entre elas, citam-se a seguir as mais significativas para este caso:

- Melhor tratamento de grandes volumes de dados (grafos de proveniência de dados de projetos de bioinformática podem conter enormes quantidades de informações);
- Maior confiabilidade quanto ao acesso simultâneo aos dados;
- Maior segurança quanto às transações assegurarem a integridade dos dados e do banco de dados em si quando os dados são alterados;
- Rapidez no acesso aos dados;
- Não é necessária a duplicação dos dados, ou seja, cada usuário, em seu computador, acessa exatamente o mesmo dado que os outros.

3.7.2 Portabilidade

O sistema criado por Paula et al é restrito a computadores com sistema operacional *Windows*. Além do mais, cada usuário deve instalar localmente o programa para poder executá-lo. O trabalho aqui descrito, por ser um sistema *web*, uma vez instalado em um servidor, permite o acesso e a utilização do sistema por qualquer usuário, com qualquer tipo de sistema operacional. Este é um dos grandes avanços obtidos com este trabalho em relação ao anterior; oferece possibilidade de um maior número de usuários terem acesso ao sistema.

3.7.3 Exportação dos Dados

No trabalho utilizado como base, pelo fato de os dados serem persistidos em arquivos XML, facilita sua utilização em outros programas semelhantes, seja para geração de grafos ou para análise dos próprios. Escolher a persistência em banco de dados em vez de em arquivo XML deixaria o presente trabalho com um ponto negativo em relação ao anterior. Porém, levando em consideração essa possibilidade, foi criada a opção de geração de arquivo XML a partir das informações contidas no banco de dados. Com essa opção, o sistema continua com as vantagens da utilização de banco de dados e ainda goza da capacidade de utilização dos dados em outros programas.

Por ser voltado para o meio acadêmico e, principalmente, para a bioinformática, a capacidade de integração com outros programas é um grande avanço. Muitas instituições criam seus próprios programas, para que possam ter exatamente as funcionalidades necessárias para cada um, e assim, a utilização dos dados de outros programas torna-se

muito complexa. Com a possibilidade da exportação dos dados para um arquivo XML, essa utilização pode ser facilitada.

3.7.4 Importação de Dados

O caminho inverso do descrito no item anterior também pode ser conseguido por meio deste sistema. Caso alguém utilize um WfMS e queira usar o ProvBioUnB como gerador do grafo de proveniência dos dados do *workflow* em questão, isso é possível. Primeiramente, cria-se um novo projeto no sistema. Posteriormente, basta fazer com que o sistema gerenciador de *workflow* insira os dados de proveniência gerados no banco de dados do ProvBioUnB. Depois de inseridos, uma vez escolhido o projeto recentemente criado, clicando na opção gerar grafo, o mesmo será gerado a partir dos dados do WfMS.

Capítulo 4

Conclusão

Tendo como base o trabalho de Paula et al. a finalidade deste projeto é uma implementação de um modelo de proveniência já consolidado, fazendo o armazenamento das informações deste modelo em um banco de dados. O modelo de proveniência escolhido foi o PROV-DM, que possui os requisitos necessários para aplicação em projetos de bioinformática. Alguns dos principais requisitos são a capacidade de representar a proveniência de um dado, descrevendo os processos e insumos utilizados em sua geração; uma representação gráfica adequada; e símbolo para representar grandes conjuntos de dados. Dois outros fatores relevantes para a escolha do PROV-DM foram a grande quantidade de documentação sobre o modelo, facilitando seu uso, e o fato de estar sendo desenvolvido pelo W3C, gerando uma possibilidade de tornar-se um padrão adotado.

Para a implementação deste trabalho foi necessário um estudo detalhado da ferramenta criada por de Paula et al., bem como do modelo PROV-DM. O objetivo final do trabalho foi a criação da ferramenta *web* ProvBioUnB. As funcionalidades implementadas foram:

- Criação de diferentes projetos;
- Inserção de entidades, atividades, agentes, coleções e relações;
- Listagem de entidades, atividades, agentes, coleções e relações;
- Edição de entidades, atividades, agentes, coleções e relações;
- Deleção de entidades, atividades, agentes, coleções e relações;
- Criação da imagem de um grafo específico de cada projeto a partir das informações contidas no banco de dados;
- Criação de um arquivo XML específico de cada projeto a partir das informações contidas no banco de dados;

4.1 Trabalhos futuros

Ainda há muito a ser feito a partir do sistema ProvBioUnB. É um sistema que foi construído de forma a permitir a fácil inclusão de novas funcionalidades, sem que as antigas sejam afetadas. Certamente, algumas funcionalidades ainda precisam ser implementadas

para que o ProvBioUnB possa ser o único programa utilizado pelos acadêmicos da bioinformática para a criação de *workflows* científicos, análise dos dados gerados a partir destes *workflows* e da proveniência dos dados neles inseridos.

O protótipo prevê um controle de permissões para os usuários. Para essa funcionalidade, deve existir um usuário *master*, autorizado a realizar todas as operações do sistema, enquanto outros usuários devem ter suas permissões definidas pelo usuário *master*. Posteriormente, as próprias permissões poderão ser gerenciadas pelo usuário *master*. Nesse sentido, podem-se definir vários tipos de usuários, entre eles, aqueles usuários com permissão de gerar grafos,e aquele apenas com permissão para visualizar grafos, sem poder alterar nenhum dado.

Como o cadastro de programas já está pronto e funcionando, a principal funcionalidade que deve ser implementada em trabalhos futuros é a de criação de um *workflow* a partir dos programas cadastrados. O usuário poderá selecionar dentre os programas cadastrados quais ele gostaria de incluir no *workflow* e quais são as relações entre os programas inseridos. Ainda na fase de construção do *workflow*, o usuário deve ser capaz de incluir arquivos de entrada para cada programa.

Importa observar que, após a criação do *workflow*, o ProvBioUnB deve ter a funcionalidade de executá-lo, com capacidade de tratamento de erros, e criação do grafo de proveniência dos dados do *workflow*.

Por fim, o sistema foi construído para suportar integrações com outros sistemas. Como o sistema gera o grafo de proveniência a partir dos dados persistidos, algum outro programa pode popular o banco de dados a partir da sua própria execução do *workflow*. Para isso, deve ser criado algum tipo de *plugin* para ser instalado nos WfMS mais comuns, que irá popular o banco do ProvBioUnB e, assim, o programa fará a geração do grafo de proveniência de um *workflow* externo.

Referências

[1] The DOT Language. http://www.graphviz.org/content/dot-language. Acessado em: 06/01/2013. 27

[2] U. Acar, P. Buneman, J. Cheney, J. Van Den Bussche, N. Kwasnikowska, and S. Vansummeren. A graph model of data and workflow provenance. In *Proceedings of the 2nd conference on Theory and practice of provenance*, TAPP'10, pages 8–8, Berkeley, CA, USA, 2010. USENIX Association. 1

[3] K. Belhajjame, R. B'Far, J. Cheney, S. Coppens, S. Cresswell, Y. Gil, P. Groth, G. Klyne, T. Lebo, J. McCusker, S. Miles, J. Myers, S. Sahoo, and C. Tilmes. Prov-dm: The prov data model. Technical report, 2012. vii, 13, 14

[4] K. Belhajjame, H. Deus, D. Garijo, G. Klyne, P. Missier, S. Soiland-Reyes, and S. Zednik. PROV Model Primer. http://www.w3.org/TR/2013/WD-prov-primer-20130312/, Março 2013. Acessado em: 20/12/2012. 14

[5] P. Buneman, S. Khanna, and W. Tan. Why and where: A characterization of data provenance. In *In ICDT*, pages 316–330. Springer, 2001. 8

[6] J. Cheney, L. Moreau, and P. Missier. Constraints of the PROV Data Model. http://www.w3.org/TR/2013/PR-prov-constraints-20130312/, Março 2013. Acessado em: 20/12/2012. 14

[7] B. Clifford, I. Foster, J. Voeckler, M. Wilde, and Y. Zhao. Tracking provenance in a virtual data grid. *Concurr. Comput. : Pract. Exper.*, 20(5):565–575, 2008. 8

[8] J. Cohen. Bioinformatics - an introduction for computer scientists. *ACM Computing Surveys*, 36:122–158, 2004. 1

[9] S. Davidson and J. Freire. Provenance and scientific workflows: challenges and opportunities. In *In Proceedings of ACM SIGMOD*, pages 1345–1350, 2008. 1

[10] E. Deelman, S. Callaghan, E. Field, H. Francoeur, R. Graves, V. Gupta, T. H. Jordan, C. Kesselman, P. Maechling, G. Mehta, D. Okaya, K. Vahi, and L. Zhao. Managing large-scale workflow execution from resource provisioning to provenance tracking: The cybershake example. In *In Proceedings of the Second IEEE international Conference on E-Science and Grid Computing*, pages 4–6, 2006. 6

[11] L. A. Gomes, S. Lifschitz, P. Picouet, P. V. S. Z. Capriles, and L. E. Dardenne. A Provenance Model for Bioinformatics Workflows. In *Brazilian Symposium on Bioinformatics 2010 Poster Proceedings*, pages 19–22, Búzios, Rio de Janeiro, Brazil, 2010. Brazilian Computer Society – SBC. 1

[12] O. Hartig and J. Zhao. Publishing and consuming provenance metadata on the web of linked data. In *IPAW*, pages 78–90, 2010. vii, 10

[13] D. Hollingsworth. WfMC: Workflow Reference Model. Technical report, Workflow Management Coalition, 1995. TC00-1003. 4, 6

[14] G. Juve and E. Deelman. Scientific workflows and clouds. *ACM Crossroads*, 16(3):14–18, 2010. 5

[15] T. Lebo, S. Sahoo, and D. McGuinness. PROV-O: The PROV Ontology. http://www.w3.org/TR/2013/PR-prov-o-20130312/, Março 2013. Acessado em: 20/12/2012. 14

[16] C. Lin, S. Lu, X. Fei, A. Chebotko, D. Pai, Z. Lai, F. Fotouhi, and J. Hua. A Reference Architecture for Scientific Workflow Management Systems and the VIEW SOA Solution. *IEEE Trans. Serv. Comput.*, 2(1):79–92, Janeiro 2009. 6, 7

[17] B. Ludäscher, M. Weske, T. Mcphillips, and S. Bowers. Scientific workflows: Business as usual? In *Proceedings of the 7th International Conference on Business Process Management*, BPM '09, pages 31–47. Springer-Verlag, 2009. vii, 4, 5, 6

[18] Tony Marston. The architecture of Web applications. http://uwasnet.org/Admin/pdf/mvc.pdf, Maio 2004. vii, viii, 17, 18

[19] L. Moreau. PROV-XML: The PROV XML Schema. http://www.w3.org/TR/2013/WD-prov-xml-20130312/, Março 2013. Acessado em: 20/12/2012. 15, 27

[20] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, B. Plale, Y. Simmhan, E. Stephan, and J. V. den Bussche. The open provenance model core specification (v1.1). *Future Gener. Comput. Syst.*, 27(6):743–756, 2011. vii, 11, 12, 13

[21] L. Moreau, O. Hartig, Y. Simmhan, J. Myers, T. Lebo, K. Belhajjame, S. Miles, and S. Soiland-Reyes. PROV-AQ: Provenance Access and Query. http://www.w3.org/TR/2013/WD-prov-aq-20130312/, Março 2013. Acessado em: 20/12/2012. 14

[22] L. Moreau and P. Missier. PROV-DM: The PROV Data Model. http://www.w3.org/TR/prov-dm/, Março 2013. Acessado em: 30/11/2012. 14

[23] L. Moreau and P. Missier. PROV-N: The Provenance Notation. http://www.w3.org/TR/2013/PR-prov-n-20130312/, Março 2013. Acessado em: 20/12/2012. 14

[24] T. Nies. Semantics of the PROV Data Model. http://www.w3.org/TR/2013/WD-prov-sem-20130312/, Março 2013. Acessado em: 20/12/2012. 15

[25] R. Paula, M. T. Holanda, M. E. M. T. Walter, and S. Lifschitz. Managing data provenance in genome project workflows. 2012. viii, 11, 14, 16, 20

[26] S. Ram and J. Liu. Active conceptual modeling of learning. chapter Understanding the semantics of data provenance to support active conceptual modeling, pages 17–29. Springer-Verlag, Berlin, Heidelberg, 2007. vii, 9

[27] Satya S. Sahoo, D. Brent Weatherly, Raghava Mutharaju, Pramod Anantharam, Amit Sheth, and Rick L. Tarleton. Ontology-driven provenance management in escience: An application in parasite research. In *Proceedings of the Confederated International Conferences, CoopIS, DOA, IS, and ODBASE 2009 on On the Move to Meaningful Internet Systems: Part II*, OTM '09, pages 992–1009, Berlin, Heidelberg, 2009. Springer-Verlag. vii, 11

[28] M. P. Singh and M. A. Vouk. Scientific Workflows: Scientific Computing Meets Transactional Workflows. *NSF Workshop*, pages 77–89, 1996. 4

[29] I. J. Taylor, M. S. Shields, I. Wang, and R. Philp. Distributed P2P Computing within Triana: A Galaxy Visualization Test Case. In *17th International Parallel and Distributed Processing Symposium (IPDPS 2003)*, pages 16–27. IEEE Computer Society, 2003. 6

[30] J. Wainer, M. Weske, G. Vossen, and C. B. Medeiros. Scientific workflow systems. *NSF Workshop on Workflow and Process Automation*, 1997. 6